SED

Tapani Tarvainen

16.10.1989

1 Johdanto

SED (*Stream EDitor*) on ei-interaktiivinen (eräajo) editori. Se on hyödyllisin komentojonoissa, erityisesti putkitettaessa kun ensimmäisen ohjelman tulostus on melkein sellaisenaan sopiva toisen syötöksi, formaatti vain on hiukan väärä (kentät väärässä järjestyksessä tms).

Sed on toiminnaltaan rivipohjainen: se lukee yhden rivin työmuistiin (*pattern space*), käsittelee sen (soveltaa siihen annettuja editointikomentoja) ja tulostaa mitä jäi jäljelle, lukee seuraavan rivin jne. (Useamman rivin käsittely yhtaikaa, rivien siirtely jne. on mahdollista mutta vaatii erityistoimenpiteitä.)

Sed on erityisen kätevä paikallisluontoisten muutosten teossa. Globaaleihin operaatioihin, kuten suurten lohkojen siirtelyyn paikasta toiseen se ei niinkään sovellu. Yleensäkin haluttaessa muuttaa tiedostoa "paikalla", saattaa paremmin soveltua normaali editori (esim. *ed*(1)), joka lukee tiedoston muistiin editoitavaksi ja lopuksi kirjoittaa sen takaisin. (Useimmat editorit osaavat lukea komentoja standard inputista, jonka voi uudelleensuunnata. Niiden käyttö filttereinä ei sen sijaan yleensä onnistu.)

Sed ei osaa laskea mitään, se tuntee vain kaikkein alkeellisimmat kontrollirakenteet jne. Jos sed-ohjelma uhkaa paisua satariviseksi kannattaa harkita työkalun vaihtoa: varsinaisten ohjelmointikielten ohella usein tulee kyseeseen *awk*(1) tai *perl* (joka ei valitettavasti ole UNIXin standardikomento).

Sedistä (kuten UNIXistakin) on valitettavasti olemassa useita ei-aivan-yhteensopivia versioita; seuraavassa on enimmäkseen rajoituttu niiden suurimpaan yhteiseen osajoukkoon (poikkeuksista on mainittu erikseen).

2 sed-komento

Sed-komennon syntaksi on:
`sed [-n] [-e` *script*...`] [-f` *sfile*...`] [`*file*...`]`

missä

- `-n` aiheuttaa sen, ettei sed tulosta kuin ne rivit, jotka erikseen käsketään tulostaa; muutoin se tulostaa kaikki rivit, joiden tulostamista ei erikseen kielletä. Saman vaikutuksen saa aikaan myös `#n` ensimmäisen komennon alussa. Esimerkeissä oletetaan ettei `-n`:ää ole käytetty, ellei erikseen toisin mainita.

- `-e` *script* ilmoittaa, että merkkijono *script* sisältää editointikomennon. (Jos `-e`-optioita on vain yksi (eikä yhtään `-f`:ää), voi "`-e`":n jättää pois, ts. `sed '`*komento*`'` on sama kuin `sed -e '`*komento*`'`.)
- `-f` *sfile* ilmoittaa, että tiedosto *sfile* sisältää yhden tai useampia editointikomentoja.
- Optioiden jälkeen seuraa lista tiedostonimiä. Jos se on tyhjä, syötöksi otetaan standard input (jota vastaa myös tiedostonimi "`-`"). Jos tiedostoja on useita, ne käsitellään niin kuin ne olisi katenoitu yhdeksi (erityisesti myös rivinumerointi jatkuu tiedostosta toiseen).

Jos `-e` ja `-f` optioita on useita, ne yhdistetään järjestyksessä yhdeksi komentojonoksi. Komentojen kokonaismäärä on rajoitettu, yläraja riippuu versiosta mutta sen pitäisi olla sata tai suurempi.

Optioita *ei* voi yhdistää, esim. `-nf` ei toimi (toimii Gnu sedissä).

Tulos tulee normaalisti stdoutiin.

3 Osoitteet

Useimmilla komennoilla voi olla nolla, yksi tai kaksi osoitetta, jotka määräävät mihin riveihin ko. komentoa sovelletaan. Osoitteita on kahdenlaisia:

- *Numeerisia*, ts. rivinumeroita (1:stä alkaen); erikoistapauksena \$ tarkoittaa viimeistä riviä.
- *Sisältöosoitteita*, ts. viitataan riveihin jotka sisältävät tietynlaisen merkkijonon. Näiden syntaksi on /*merkkijonolauseke*/ tai \\`x`*merkkijonolauseke*`x`, missä `x` on mielivaltainen merkki. (Merkkijonolausekkeista myöhemmin enemmän.)

Osoitteet tulkitaan seuraavasti:

- Jos osoitetta ei ole lainkaan: Käskyä sovelletaan kaikkiin riveihin.
- Yksi osoite: Käsky viittaa riveihin joita ko. osoite vastaa, numeerinen osoite täsmälleen ko. riviin, sisältöosoite kaikkiin riveihin, joilla annetunlainen merkkijono esiintyy.
- Kaksi osoitetta: Käsky suoritetaan alkaen ensimmäisen osoitteen osoittamasta rivistä ja päättyen jälkimmäisen määräämään riviin (molemmat mukaanluettuna).

Jos osoitteita on kaksi, ne voivat molemmat olla numeerisia tai sisältöosoitteita. Eri tapaukset tulkitaan seuraavasti:

- n_1, n_2: Rivit n_1–n_2. Jos $n_1 > n_2$ käskyä ei suoriteta koskaan (ilman virheilmoitusta).

- n_1,*lauseke*$_2$: Kaikki rivit n_1:stä alkaen ensimmäiseen sitä seuraavaan, jolla *lauseke*$_2$ esiintyy, tai tiedoston loppuun ellei sellaista löydy.
- *lauseke*$_1$,n_2: Alkaen ensimmäisestä rivistä, jolla *lauseke*$_1$ esiintyy, riville n_2, tai tiedoston loppuun jos *lauseke*$_1$ esiintyy ensimmäisen kerran rivin n_2 jälkeen.
- *lauseke*$_1$,*lauseke*$_2$: Alkaen riviltä, jolla *lauseke*$_1$ esiintyy, sitä lähinnä seuraavaan jolla *lauseke*$_2$ esiintyy, ja jos sen jälkeen esiintyy taas *lauseke*$_1$, jälleen siitä seuraavaan *lauseke*$_2$-riviin, jne.

Huom. Jos toinen osoite on sisältöosoite, käsky kohdistuu aina vähintään kahteen riviin (ellei tiedosto lopu ensin).

Osoitteiden merkityksen voi kääntää niiden perään sijoitetulla huutomerkillä (käskyä sovelletaan kaikkiin riveihin *paitsi* ...).

4 Merkkijonolausekkeet (*regular expressions*)

Sisältöosoitteissa ja korvaa-komennossa käytettävät merkkijonolausekkeet tunnistavat joukon erikoismerkkejä; tarkalleen minkä joukon riippuu versiosta, mutta ainakin seuraavien pitäisi kelvata kaikille:

. Edustaa yhtä mielivaltaista merkkiä, myös rivinvaihtoa työmuistin sisällä (muttei sen lopussa), ja erityisesti `.*` ei ylitä rivinvaihtoa. (Joissakin versioissa ei vastaa rivinvaihtoa koskaan.)

^ Edustaa rivin alkua (kuviteltua nollamittaista merkkiä työmuistin alussa — jos työmuistissa on useita rivejä, *ei* vastaa rivinvaihtoa sen keskellä) ollessaan lausekkeen alussa (muualla normaali merkki).

$ Edustaa rivin loppua (kuviteltua nollamittaista merkkiä työmuistin lopussa — ei rivinvaihtoa työmuistin keskellä) ollessaan lausekkeen lopussa (muualla normaali merkki).

\n Vastaa rivinvaihtomerkkiä työmuistin keskellä (ei sen lopussa).

***** Vastaa nollaa tai useampaa kopiota edeltävästä yhden merkin osalausekkeesta. (Joissakin versioissa toimii myös \(*lauseke*\):n perässä.)

[*merkkejä*] Vastaa mitä tahansa hakasulkujen sisällä olevista merkeistä (tasan yhtä), paitsi jos ensimmäinen merkki siellä on ^, jolloin vastaa mitä tahansa merkkiä jota *ei* ole siellä. Lisäksi - kahden merkin välissä tarkoittaa väliä, esim. `[a-z]` vastaa kaikkia kirjaimia a–z. Jos joukkoon halutaan], se on sijoitettava heti alku-[:n jälkeen. Muilla erikoismerkeillä ei ole erikoismerkitystä hakasulkujen sisällä (paitsi `\n` toimii).

\(*lauseke*\) Rajoittaa osan lausekkeesta niin että siihen voi viitata *numero*:lla.

n missä *n* on numero 1–9. Vastaa *n*:nnen sulkulausekkeen arvoa. Esim. "`\([Kk][a-zA-Z]\)* \1`" vastaa k-kirjaimella alkavaa sanaa, joka esiintyy kahdesti peräkkäin yhdellä välilyönnillä erotettuna (ja molemmilla kerroilla joko isolla tai pienellä alkukirjaimella).

**** Jos jotakin erikoismerkkiä halutaan käyttää normaalina merkkinä, sen eteen on lisättävä \ (tai sen voi sijoittaa hakasulkuihin).

Näiden lisäksi ainakin seuraavat esiintyvät joissakin versioissa:

// Tyhjä lauseke vastaa lähinnä edellistä lauseketta.

\ < Sanan alku, ts. kohta, jonka oikealla puolella on kirjain, numero tai alaviiva ja vasemmalla jotakin muuta (myös rivin alku).

\ > Sanan loppu, kohta, jonka vasemmalla puolella on kirjain, numero tai alaviiva ja oikealla jotakin muuta (myös rivin loppu).

\{m[,[n]\}] Toistaa edeltävää merkkiä (tai osalauseketta) tasan m, (\{m\}), vähintään m (\{m,\}) tai m–n (\{m,n\}) kertaa. \{0,\} on sama kuin `*`.

(seuraavat toimivat lähinnä Gnu-sedissä)

\+ tai vain +: Toista edellistä merkkiä (tai osalauseketta) yksi tai useampia kertoja. Sama kuin \{1,\}.

\? tai vain ?: Toista edellistä merkkiä (tai osalauseketta) nolla tai yksi kertaa. Sama kuin \{0,1\}. (Huom. se, että `+` ja `?` saattavat toimia \:n kanssa tai ilman aiheuttaa ongelmia kun niiden halutaan esiintyvän normaaleina merkkeinä: Silloin ei auta muu kuin selvittää miten ne toimivat käsillä olevassa versiossa, tai panna ne hakasulkuihin.)

\| "tai"; vastaa oikealla *tai* vasemmalla puolellaan olevaa osalauseketta. Esim. `kissa\|koira` ja `k\(iss\|oir\)a` vastaavat sanoja "kissa" ja "koira".

\w Sama kuin `[a-zA-Z0-9]`.

\W Sama kuin `[^a-zA-Z0-9]`.

\b Sanan raja, ts. kohta, jonka toisella puolella on kirjain tai numero ja toisella ei (kummin päin tahansa).

\B Sanan sisällä tai ulkopuolella, ts. kohta, jonka molemmilla puolilla joko on kirjain tai numero tai kummallakaan ei ole.

Jos lauseke täsmää tekstiin useammalla tavalla, sääntö on se, että se vastaa niistä vasemmanpuoleisinta, ja jos niitäkin on useita, pisintä.

5 Editointikomennot

5.1 Yleistä

Komentojen syntaksi on seuraavaa muotoa:

[osoite1[,osoite2]]käsky[argumentit]

Seuraavassa osoitteita ei ole merkitty näkyviin; käskyt hyväksyvät 0–2 osoitetta ellei toisin ole mainittu (jotkut käskyt hyväksyvät vain yhden osoitteen, jotkut eivät yhtään).

Yleensä jokainen käsky on omalla rivillään; niitä voi myös sijoittaa samalle riville puolipisteellä erotettuna (ei välttämättä toimi kaikissa versioissa). Jokainen erillinen `-e`-optio vastaa yhtä riviä.

5.2 Kokorivikomennot

d (*delete*): Poista rivejä. Suoritus jatkuu uuden rivin lukemisen jälkeen ensimmäisestä komennosta.

n (*next line*): Tulosta työmuisti (paitsi jos `-n` optio on voimassa) ja lue seuraava rivi. Suoritus jatkuu ensimmäisestä komennosta.

a (*append*): Tulosta teksti käsiteltävän rivin jälkeen. Lisättävä teksti on komentoa seuraavilla riveillä, ja jos sitä on useampia rivejä on kaikkien paitsi viimeisen loppuun lisättävä \\. Enintään yksi osoite.

i (*insert*): Kuten **a**, mutta teksti lisätään *ennen* käsiteltävää riviä.

c (*change*): Kuten **a** ja **i**, mutta teksti *korvaa* ko. rivit.

Jos osoitteita on kaksi, ko. rivit poistetaan ja korvataan yhdellä kopiolla tekstistä.

Jos osoitteita on yksi, jokainen vastaava rivi korvataan tekstillä.

Jos osoitteita ei ole yhtään, koko tiedosto korvataan tekstillä.

Esimerkkejä:

- Poista rivit, jotka alkavat sanalla `REM`, isoilla tai pienillä kirjaimilla kirjoitettuna:

```
/^[Rr][Ee][Mm]/d
```

- Lisää tiedoston loppuun rivi `*** The End ***`:

```
$a\
*** The End ***
```

- Korvaa kolme ensimmäistä riviä rivillä `ALKU`:

```
1,3c\
ALKU
```

5.3 I/O-komennot

Nämä komennot eivät vaikuta työmuistin sisältöön, ainoastaan tulostukseen.

p (*print*): Tulosta työmuistin sisältö stdoutiin. Huom. työmuistin sisältö tulostetaan normaalisti automaattisesti käsittelyn lopuksi, ellei `-n`-optiota ole käytetty; `sed 'p'` tulostaisi jokaisen rivin kahdesti.

l (*list*): Kuten **p**, mutta kontrollimerkit korvataan *nnn* oktaali-notaatiolla ja yli 72 merkkiä pitkät rivit katkaistaan. Rivin sisällä olevat \\nt merkitään \\illa rivin lopussa.

w ⟨*tiedosto*⟩ (*write to file*): Kuten **p**, mutta tulosta tiedostoon ⟨*tiedosto*⟩. **w**:n ja tiedostonimen välissä on oltava tasan yksi välilyönti.

r ⟨*tiedosto*⟩ (*read from file*): Kuten **a**, mutta lisättävä teksti luetaan tiedostosta ⟨*tiedosto*⟩. **r**:n ja tiedostonimen välissä on oltava tasan yksi välilyönti.

Jos komentojonossa on useita **r** ja **a** komentoja, niiden lisäämät tekstit tulostetaan samassa järjestyksessä kuin komennot esiintyvät, sen jälkeen kun kaikki muut nykyriviin vaikuttavat komennot on suoritettu.

5.4 Korvaa-komento

Korvaa (*substitute*) komennon syntaksi on seuraava:

`s/`⟨*merkkijonolauseke*⟩`/`*korvaava teksti*`/`⟨*liput*⟩

minkä edessä voi taas olla 0–2 osoitetta.

Korvattavan merkkijonon (⟨*merkkijonolauseke*⟩) rajoittimina voi käyttää kauttaviivan asemesta mitä tahansa merkkiä paitsi välilyöntiä tai `\n`:ää.

Korvaava teksti voi sisältää mitä tahansa. Merkkijonolausekkeissa käytetyillä erikoismerkeillä ei ole erikoismerkitystä, ainoastaan rajoittimena käytetty merkki, \ itse, rivinvaihto sekä & (ks. alla) on suojeltava edeltävällä \:lla.

Korvaavassa tekstissä voi käyttää löydettyä tekstiä seuraavasti:

`&` edustaa merkkijonolauseketta vastaavaa tekstiä kokonaisuudessaan.

\\n , missä n on jokin numero 0–9, edustaa n:nnen sulkulausekkeen sisältöä merkkijonolausekkessa.

Liput: **s**-komento tuntee seuraavat liput:

g : Korvaa kaikki esiintymät (oletuksena vain ensimmäinen joka riviltä korvataan).

n : Korvaa vain n:s esiintymä joka rivillä, kun n on kokonaisluku 1–512.

p : Tulosta rivi stdoutiin jos ⟨merkkijonolauseke⟩ löytyi (ja korvattiin).

w ⟨*tiedosto*⟩ : Kuten **p**, mutta tulosta nimettyyn tiedostoon stdout:in asemesta.

Nämä voivat esiintyä millaisina yhdistelminä tahansa, paitsi **p** ja **w** eivät kaikissa versioissa voi esiintyä yhtaikaa (ei myöskään useita **w**-optioita).

Esimerkkejä

- Muuta jokainen "tex", "TEX" jne. muotoon "TeX":

  ```
  s/[Tt][Ee][Xx]/TeX/g
  ```

- Vaihda jokaisen rivin kahden ensimmäisen sanan järjestys, olettaen että sanat muodostuvat kirjaimista a–z ja niiden välissä on mahdollisesti pilkku, piste tai puolipiste (enintään yksi näistä) sekä välilyöntejä:

  ```
  s/^\([a-zA-Z][a-zA-Z]*\)\([,.; ] *\)\([a-zA-Z][a-zA-Z]*\)/\3\2\1/
  ```

- Lisää jokaisen "defined"-sanan sisältävän rivin alkuun kolme huutomerkkiä:

  ```
  s/^.*defined/!!!&/
  ```

  tai

  ```
  /defined/s/^/!!!/
  ```

5.5 Muunna-komento

Muunna (*transform*) komento muuttaa yksittäisiä merkkejä toisiksi. Sen syntaksi on seuraavanlainen:

`y/`*merkkijono1*`/`*merkkijono2*`/`

Jokainen *merkkijono1*:ssä esiintyvä merkki korvataan *merkkijono2*:ssa vastaavalla paikalla olevalla merkillä. *merkkijono1*:n ja *merkkijono2*:n on oltava yhtä pitkiä. Muunnosta sovelletaan kuhunkin merkkiin vain kerran.

Esim.

```
y/ab/ba/
```

vaihtaa kaikki a-kirjaimet b:ksi ja päinvastoin;

```
y/abcdefghijklmnopqrstuvwxyz/ABCDEFGHIJKLMNOPQRSTUVWXYZ/
```

muuttaa kaikki pienet kirjaimet isoiksi.

5.6 Monen rivin käsittely yhtaikaa

N Lue seuraava rivi työmuistiin sen nykyisen sisällön perään. Väliin jää yksi rivinvaihto (`\n`).

D Tuhoa työmuistin ensimmäinen rivi (alkuosa ensimmäiseen `\n`:ään saakka. Suoritus jatkuu ensimmäisestä komennosta. Jos työmuisti tyhjenee (siellä on on vain yksi `\n` lopussa), sinne luetaan seuraava rivi (toimii tällöin täsmälleen kuten **d**). Ei hyväksy yhtään osoitetta.

P Tulosta työmuistin ensimmäinen rivi (alusta ensimmäiseen `\n`:ään saakka). Ei hyväksy yhtään osoitetta.

Näitä käytetään yleensä ryhmittely- ja hyppykomentojen kanssa.

Yksinkertaisia monen rivin kuvioita voi toki etsiä muutenkin: Esim. seuraava etsii paikat, joissa sama sana esiintyy rivin lopussa ja seuraavan alussa, ja poistaa jälkimmäisen esiintymän.

```
N
s/\([a-zA-Z][A-Za-z]*\) *\n *\1/\1\n/
P
D
```

5.7 Apumuistin käyttö

Editointioperaatioihin käytettävän työmuistin lisäksi on käytettävissä erillinen apumuisti (*hold area*). Suorituksen alkaessa se on aina tyhjä.

h Kopioi työmuisti apumuistiin. Apumuistin aiempi sisältö tuhoutuu.

H Kopioi työmuisti apumuistiin sen nykyisen sisällön perään.

g Kopioi apumuisti työmuistiin. Työmuistin aiempi sisältö tuhoutuu.

G Kopioi apumuisti työmuistiin sen nykyisen sisällön perään.

x Vaihda työmuistin ja apumuistin sisällöt keskenään.

Esim. Siirrä tiedoston ensimmäinen rivi viimeiseksi:

```
1h
1d
$G
```

Käännä rivit päinvastaiseen järjestykseen (tulosta viimeinen ensimmäiseksi jne):

```
#n
G
h
$p
```

Huom. Apumuistin koko on rajoitettu (raja saattaa jopa olla melko pieni); tämäkin esimerkki todennäköisesti kaatuu, jos tiedosto on suuri.

5.8 Suorituksen ohjauskomennot

! "Älä": Käännä osoitteen merkitys, ts. seuraavaa komentoa sovelletaan kaikkiin riveihin *paitsi* edeltävää osoitetta (osoiteparia) vastaaviin.

{ } Ryhmittely: Samaa osoitetta sovelletaan joukkoon komentoja. Käytetään seuraavasti:

⟨*osoite1*⟩[, ⟨*osoite2*⟩]{[*komento1*]
komento2
...
}

Ryhmiä voi olla sisäkkäin (enintään 100 tasoa).

:⟨*nimiö*⟩ Merkitsee kohdan johon **b**- tai **t**-komennolla voi hypätä. Enintään 7 merkkiä. Ei osoitetta.

b⟨*nimiö*⟩ Ehdoton hyppy. Jos ⟨*nimiö*⟩ puuttuu, hyppää komentojonon loppuun.

t⟨*nimiö*⟩ Ehdollinen hyppy. Hyppää jos yksikin korvaa-komento on onnistuneesti korvannut jotakin rivin lukemisen tai edellisen **t**-komennon jälkeen; ellei, suoritus jatkuu seuraavasta komennosta.

Ryhmittelyä voi käyttää mm. erilaisten osoiteyhdistelmien tekoon:

```
1,10{/^REM/d
}
```

poistaa REM-alkuiset rivit 10:n ensimmäisen joukosta;

```
#n
1,30{11,20!p
}
```

tulostaa rivit 1–10 ja 21–30.

Sisältöosoitteet testaavat työmuistin senhetkistä sisältöä; **b**-komentokin voi siten itse asiassa olla ehdollinen hyppy.

5.9 Sekalaisia komentoja

= Tulosta käsiteltävänä olevan rivin numero (rivin alkuun jos tämä itse tulostetaan). Hyväksyy enintään yhden osoitteen.

q (*quit*): Lopeta editointi välittömästi. Työmuistin sisältö käsitellään kuin olisi hypätty komentojonon loppuun ja lopetetaan siihen (uutta riviä ei enää lueta). Vaatii tasan yhden osoitteen.

\# Kommentti: `#`:lla alkavat rivit (samoin tyhjät rivit) eivät vaikuta mitään. Poikkeuksena `#n` ensimmäisen rivin alussa, joka saa aikaan saman vaikutuksen kuin `-n`-optio.

6 Rajoituksia

Eri sed-versiot asettavat erilaisia rajoituksia ohjelmien koolle yms. Seuraavat rajat edustavat jonkinlaista minimiä (näihin mahtuvan pitäisi mennä läpi kaikista).

- Komentojen määrä: Enintään 100.
- Nimiöiden määrä: Enintään 48.
- Nimiöiden pituus: Enintään 7 merkkiä.
- Ryhmittely: Enintään 100 sisäkkäistä ryhmää.
- Komentorivin pituus: Enintään 4000 merkkiä.
- Työmuisti: Enintään 4000 merkkiä.
- Apumuisti: Enintään 4000 merkkiä.
- Tiedostot: **w**-komennoilla voi kirjoittaa enintään 10:een eri tiedostoon, tai 9:ään jos **r**-komentoa on käytetty.
- Alkutyhjät: Jotkut versiot poistavat välilyönnit ja tabit rivin alusta.
- Merkkijonolausekkeet: Ainoat erikoismerkit, joiden voi luottaa toimivan samalla tavalla, ovat `.` `*` `[]` `^` `$` `\(` `\)` `\`*numero* ja `\n`.

Käytettävä shell asettaa lisäksi omat rajoituksensa (komentorivin pituus on rajallinen, rivinvaihdon syöttäminen ei onnistu tms), jotka voi yleensä kiertää käyttämällä `-f`-optiota.

7 Esimerkki: C-prototyyppien muodostus

```
# PROTO.SED - build ANSI-style C function prototypes from K&R-style source
# Usage: sed -f proto.sed [file(s)]
#
# Assumes function definitions begin at column 1.
# Comment delimiters inside strings aren't handled properly,
# nor can this cope with variable argument lists,
# and of course #if's and macros can cause no end of trouble.
#
# Tapani Tarvainen 10/16/89
#
# junk initial comments
/\/\*/{
:COM
# try deleting /* ... */
s-/\*.*\*/--
tNOCOMM
# didn't work, get another line
N
# junk up to newline (comments can be loooong ...)
s-/\*.*\n-/\*-
# t below jumps always, but b wouldn't clear substitution flag
tCOM
:NOCOMM
}
# look for lines beginning with a letter ('_' counts as a letter)
/^[a-zA-Z_]/{
# look for left parenthesis; anything but letter, digit, * or
# whitespace before it means this can't be function definition
:PAREN
/^[^(]*[^a-zA-Z0-9_*  \n(]/bQUIT
/(/bBRACE
N
# check for comments within
\-/\*-bCOM2
bPAREN
:COM2
s-/\*.*\*/--
tPAREN
N
s-/\*.*\n-/\*-
tCOM2
# then read up to {
:BRACE
# ); means this is a declaration or ready prototype, don't touch it
/)[  \n]*;/bQUIT
s/{/;/
tGOTALL
N
# check for comments again
\-/\*-bCOM3
bBRACE
:COM3
\-/\*.*\*/-tBRACE
N
```

```
s-/\*.*\n-/\*-
tCOM3
:GOTALL
# we've got it all (and no comments left)
# convert newlines and tabs to spaces
s/[\n ]/ /g
# convert "int m, n;" into "int m; int n;" etc.
:DUPTYPE
s/\([);] *\)\([a-zA-Z_][a-zA-Z0-9_ ]* \)\([^,;]*\),/\1\2\3; \2/
tDUPTYPE
# match names before and after ) and move type specifiers inside parens
:MATCH
s/\((.*\)\([a-zA-Z_][a-zA-Z0-9_]*\)\(.*\))\([a-zA-Z0-9_*# ]*[* ]\2\);/\1\4\3)/
tMATCH
# prepend "int " to those still typeless
s/\([(,] *\)\([a-zA-Z_][a-zA-Z0-9_]*\)\( *[),]\)/\1int \2\3/g
# if param list is empty, insert "void"
s/(\( *\))/(\1void)/
# if function type is missing, insert "int"
s/^[a-zA-Z_][a-zA-Z0-9_]* *(/int &/
s/^static \([a-zA-Z_][a-zA-Z0-9_]* *(\)/static int \1/
# final polish: discard extra spaces &c
# (this is the place to change if you want more spaces &c in output)
s/  */ /g
s/( /(/
s/ ;/;/
# uncomment the next two lines to get each parameter to a line of its own
#s/\(,\) */\1\
# /g
p
:QUIT
}
d
#
```